CS425 Final Project_Team-T_Inventory Management System

S.No	Name	A-number	Contribution(%)
1.	Hemanth Thathireddy	A20525346	20%
2.	Satya Dineswara Reddy Satti	A20528639	20%
3.	Sai Manasa Basani	A20544372	20%
4.	Alavalapati Veera manohar Reddy	A20526070	20%
5.	Ying Zhang	A20547843	20%

3rd Deliverable (Test a variety of SQL queries)

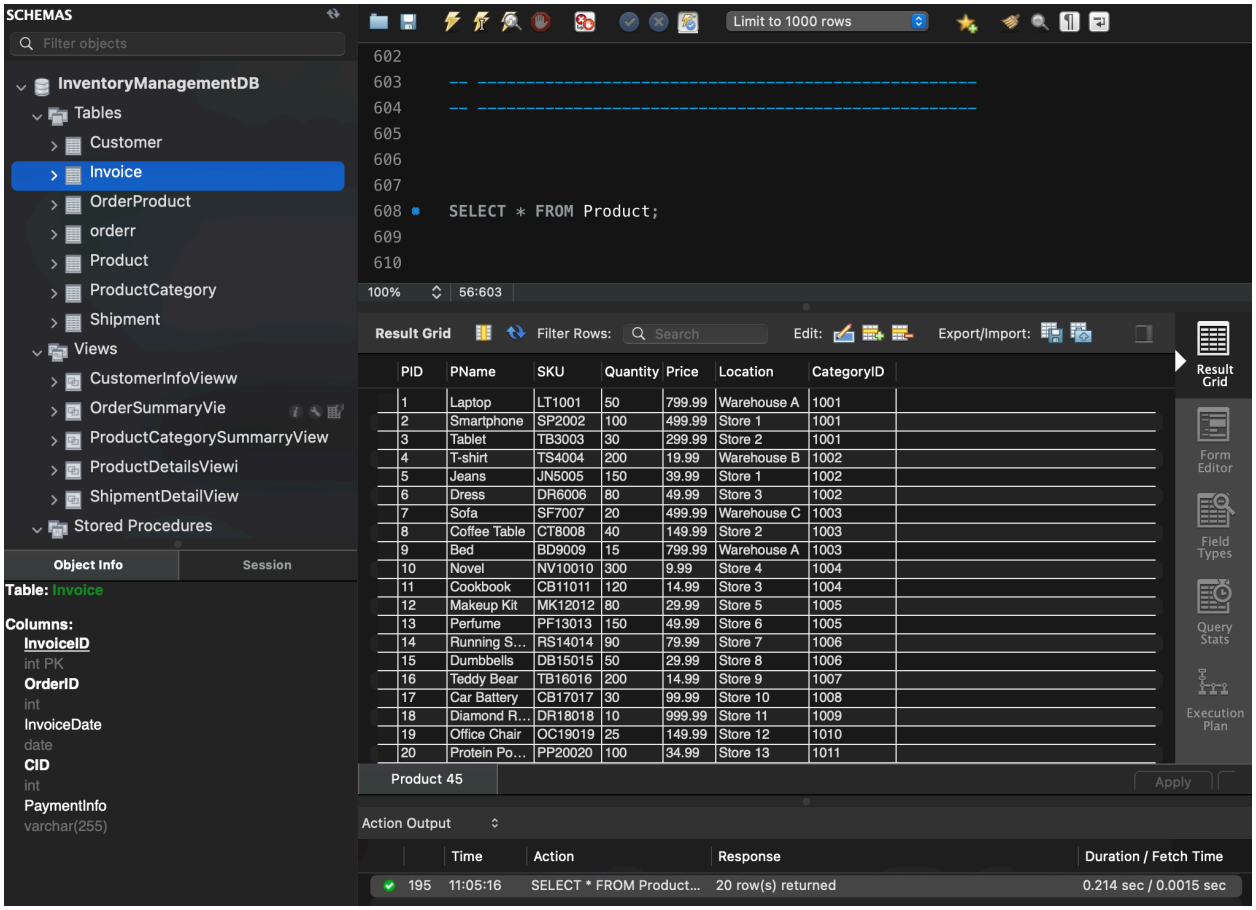
1) **Basic Select Query:**

Query:

```
SELECT * FROM Product;
```

Explanation: This query retrieves all columns from the Product table, displaying details of all available products.

Output/result:



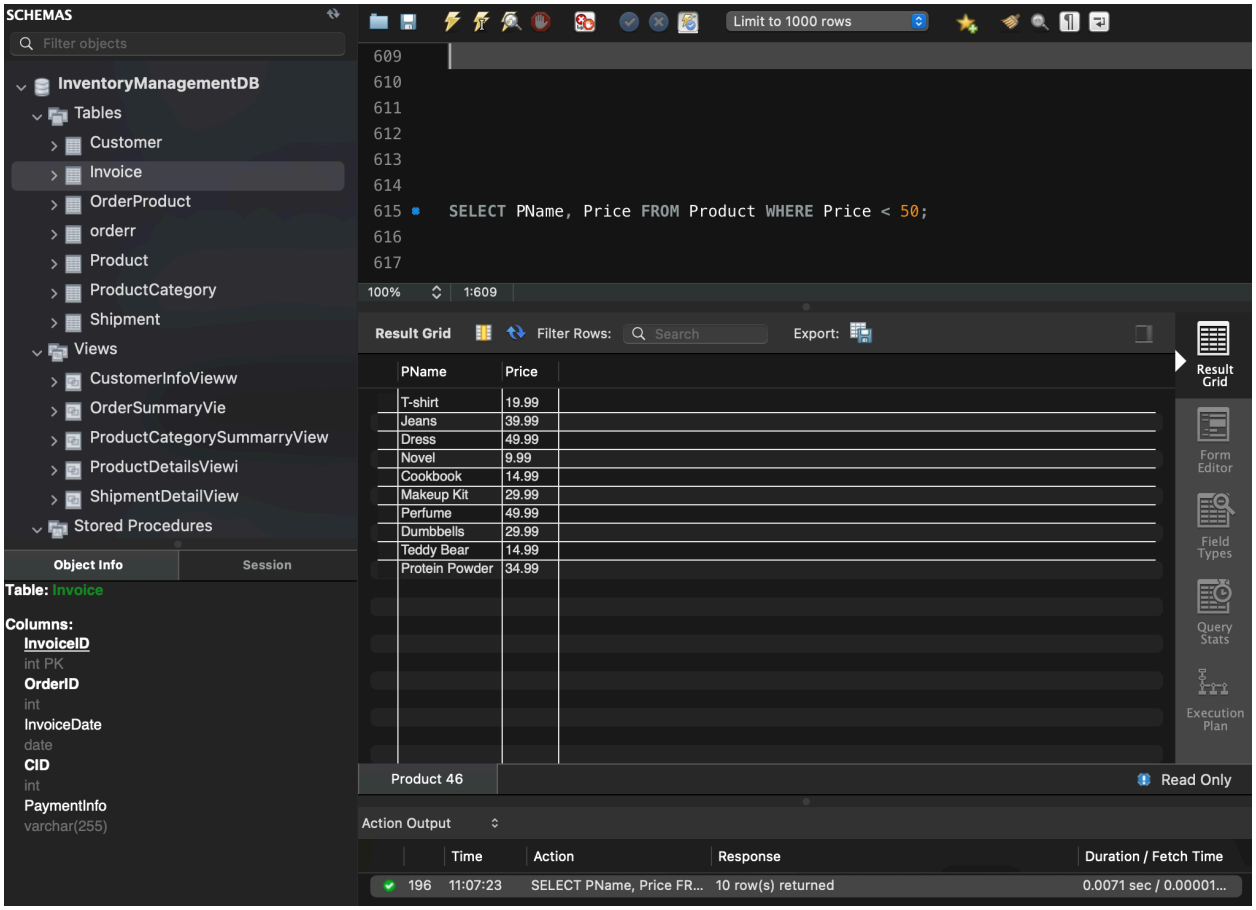
2) Filtering with WHERE Clause:

Query:

```
SELECT PName, Price FROM Product WHERE Price < 50;
```

Explanation: This query retrieves the names and prices of products with a price less than 50.

Output/result:



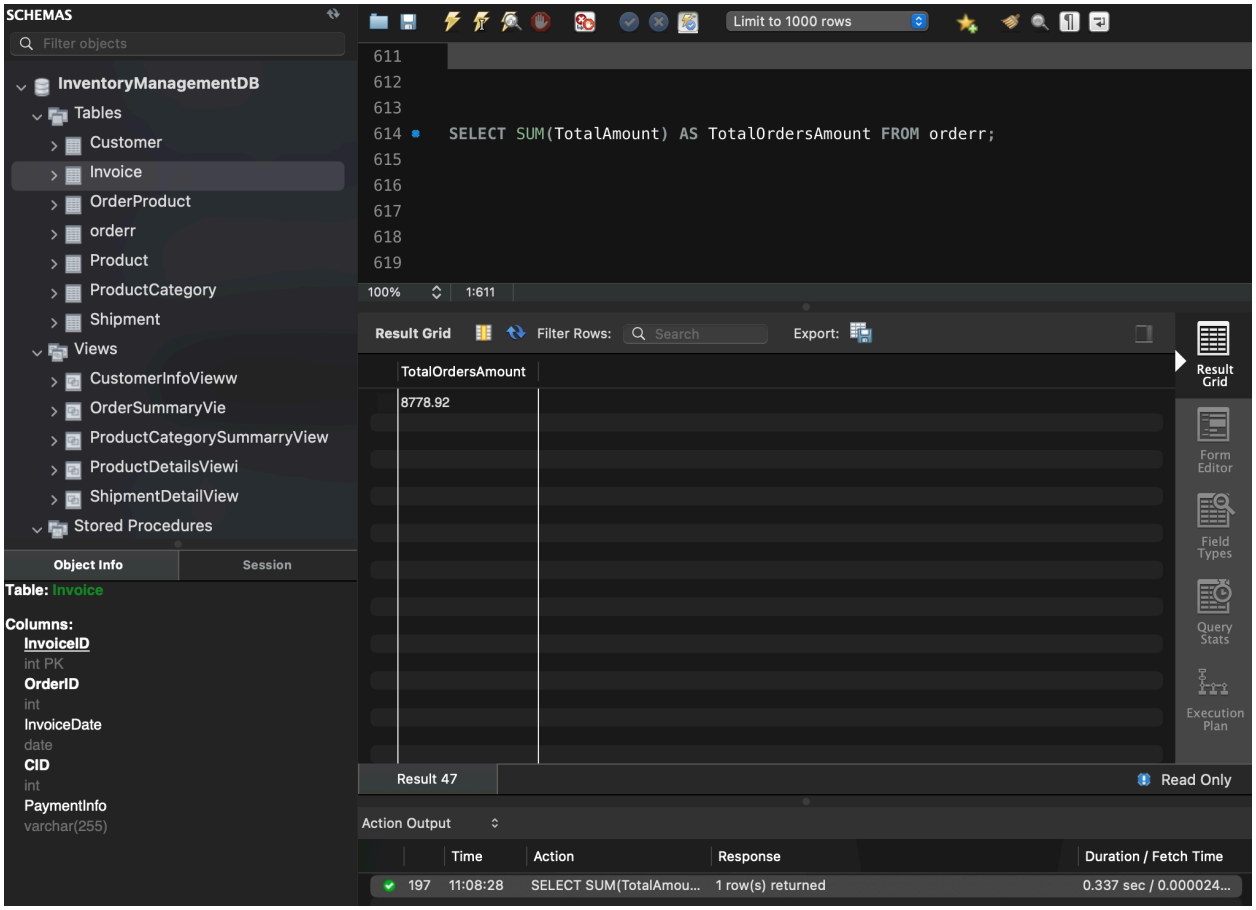
3) Aggregate Function - Total Orders Amount:

Query:

```
SELECT SUM(TotalAmount) AS TotalOrdersAmount FROM orderr;
```

Explanation: This query calculates the total amount of all orders in the 'orderr' table.

Output/result:



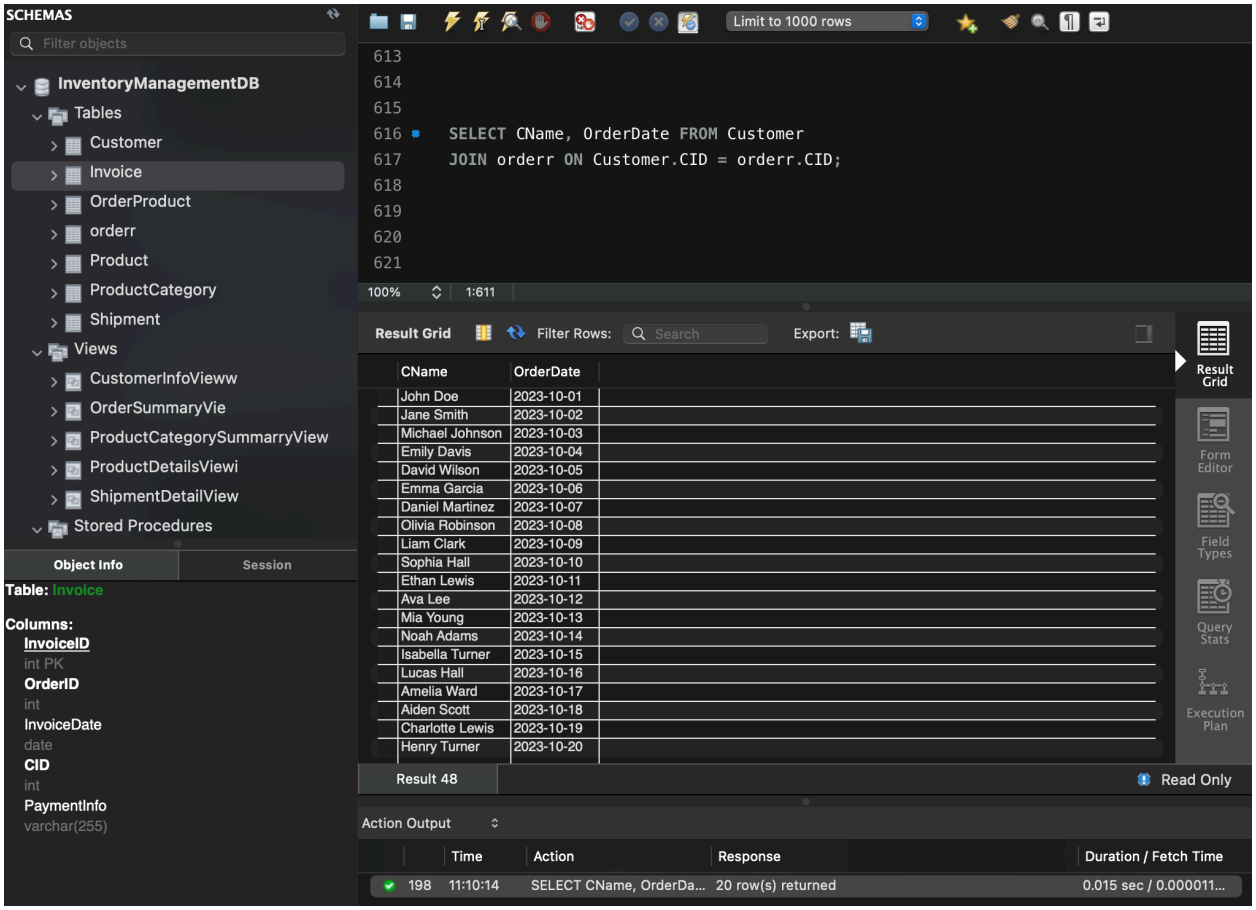
4) Joining Tables - Customer and Order:

Query:

```
SELECT CName, OrderDate FROM Customer
JOIN orderr ON Customer.CID = orderr.CID;
```

Explanation: This query joins Customer and orderr tables and retrieves customer names and corresponding order dates.

Output/result:



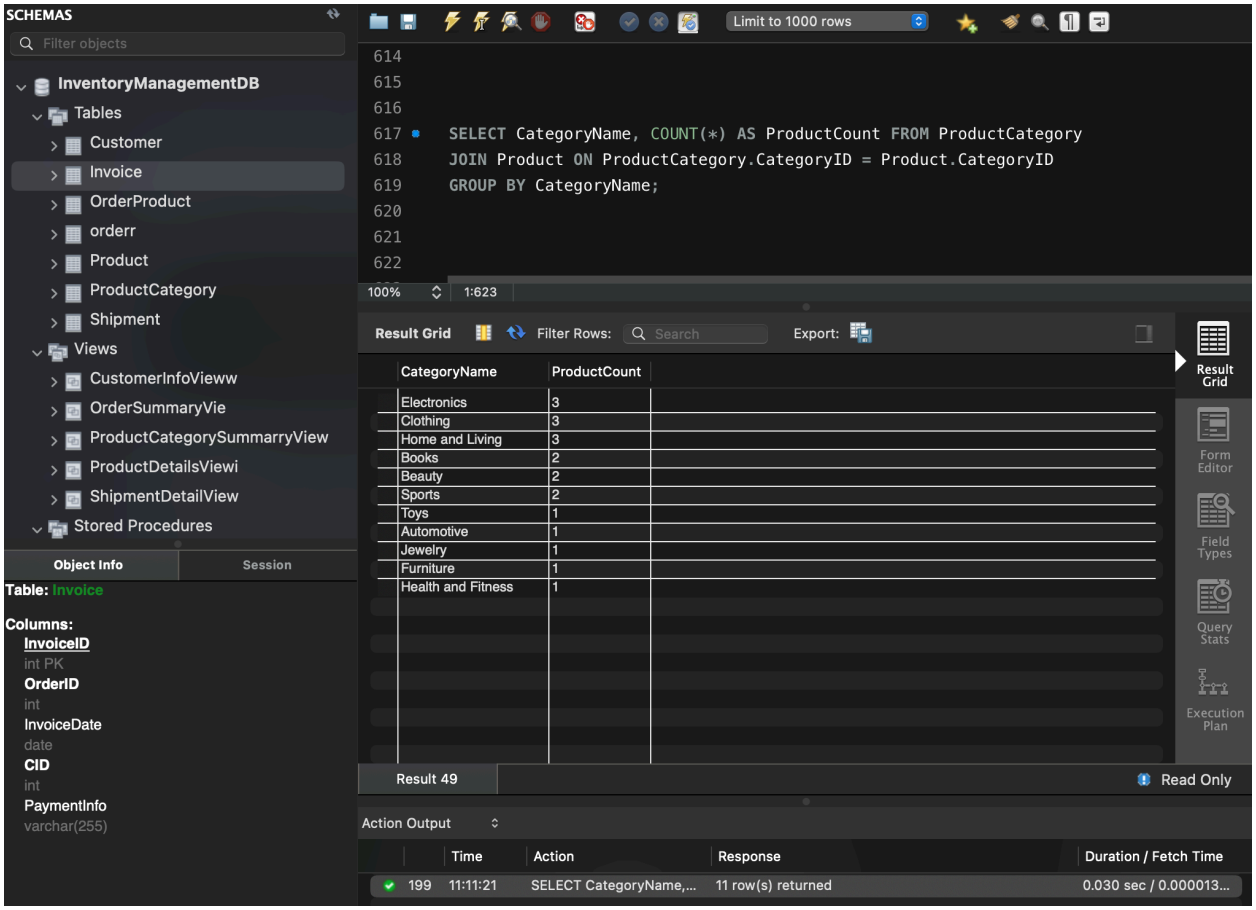
5) : Number of products in each category

Query:

```
SELECT CategoryName, COUNT(*) AS ProductCount FROM ProductCategory
JOIN Product ON ProductCategory.CategoryID = Product.CategoryID
GROUP BY CategoryName;
```

Explanation: This query counts the number of products in each category and displays the category name along with the product count.

Output/result:



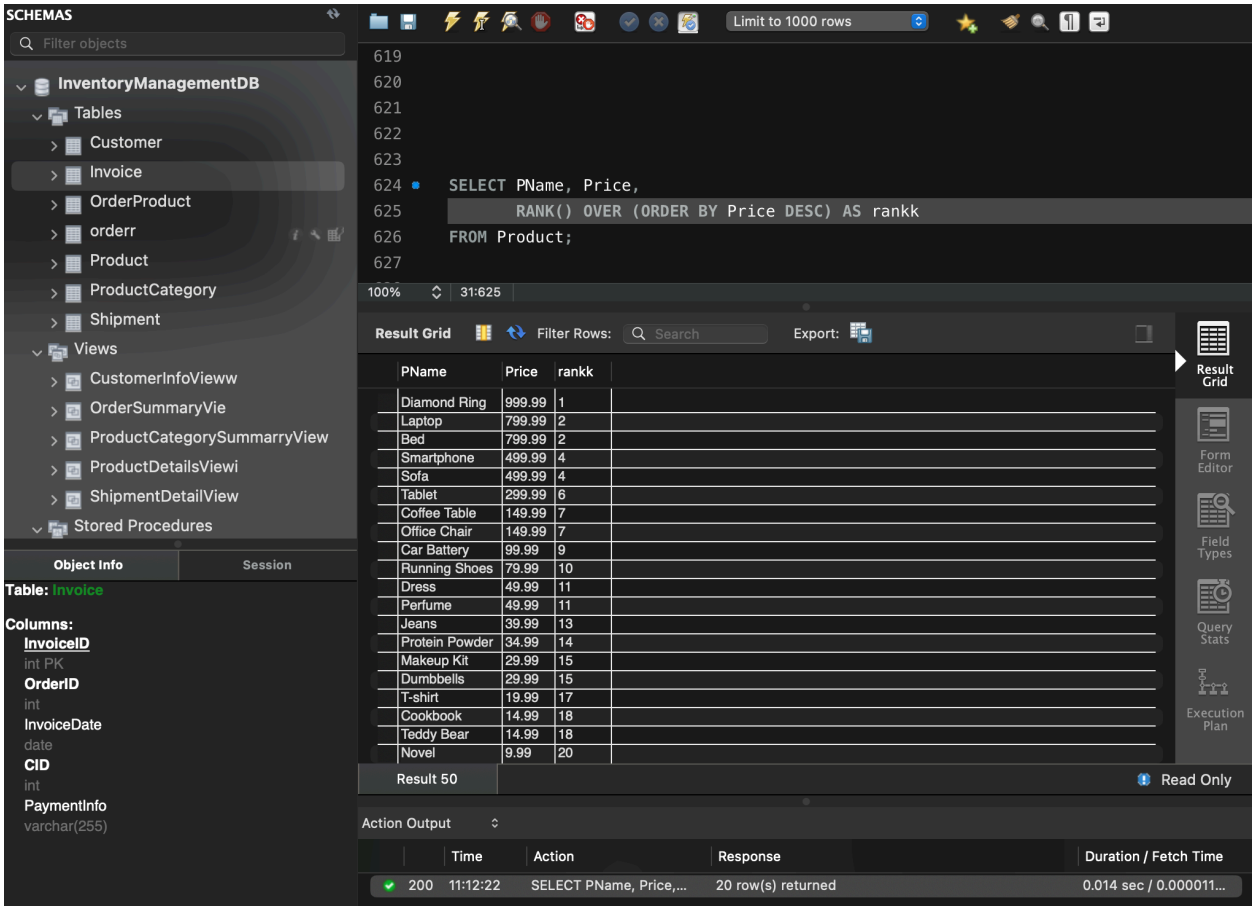
6) Window Function - Ranking Products by Price:

Query:

```
SELECT PName, Price,
    RANK() OVER (ORDER BY Price DESC) AS rankk
FROM Product;
```

Explanation: This query assigns a rank to products based on their prices, ordering them in descending order.

Output/result:

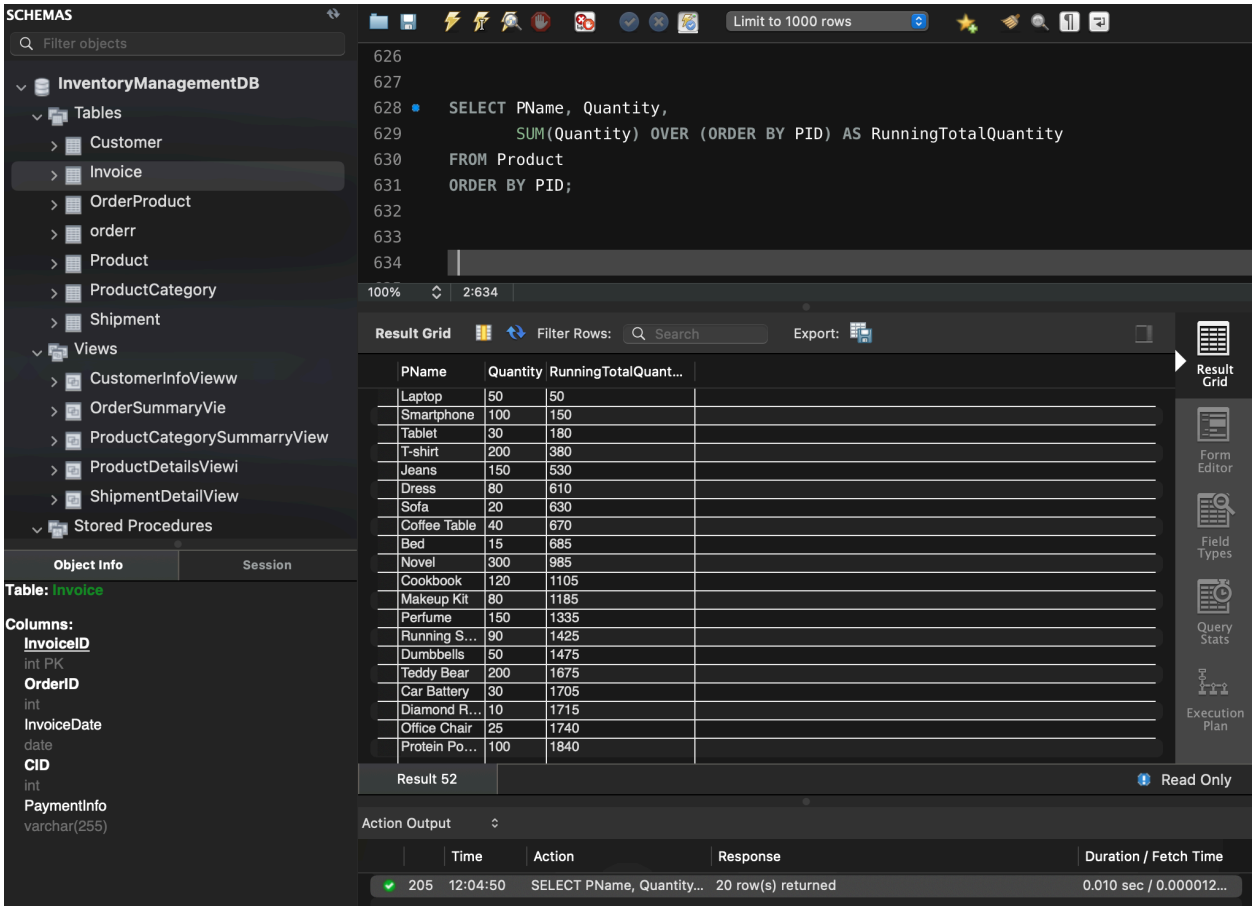


7) Calculating Running Total of Quantity Sold:

```
SELECT PName, Quantity,
    SUM(Quantity) OVER (ORDER BY PID) AS RunningTotalQuantity
FROM Product
ORDER BY PID;
```

Explanation: This query calculates the running total of quantities sold for products, ordered by their Product IDs.

Output/result:



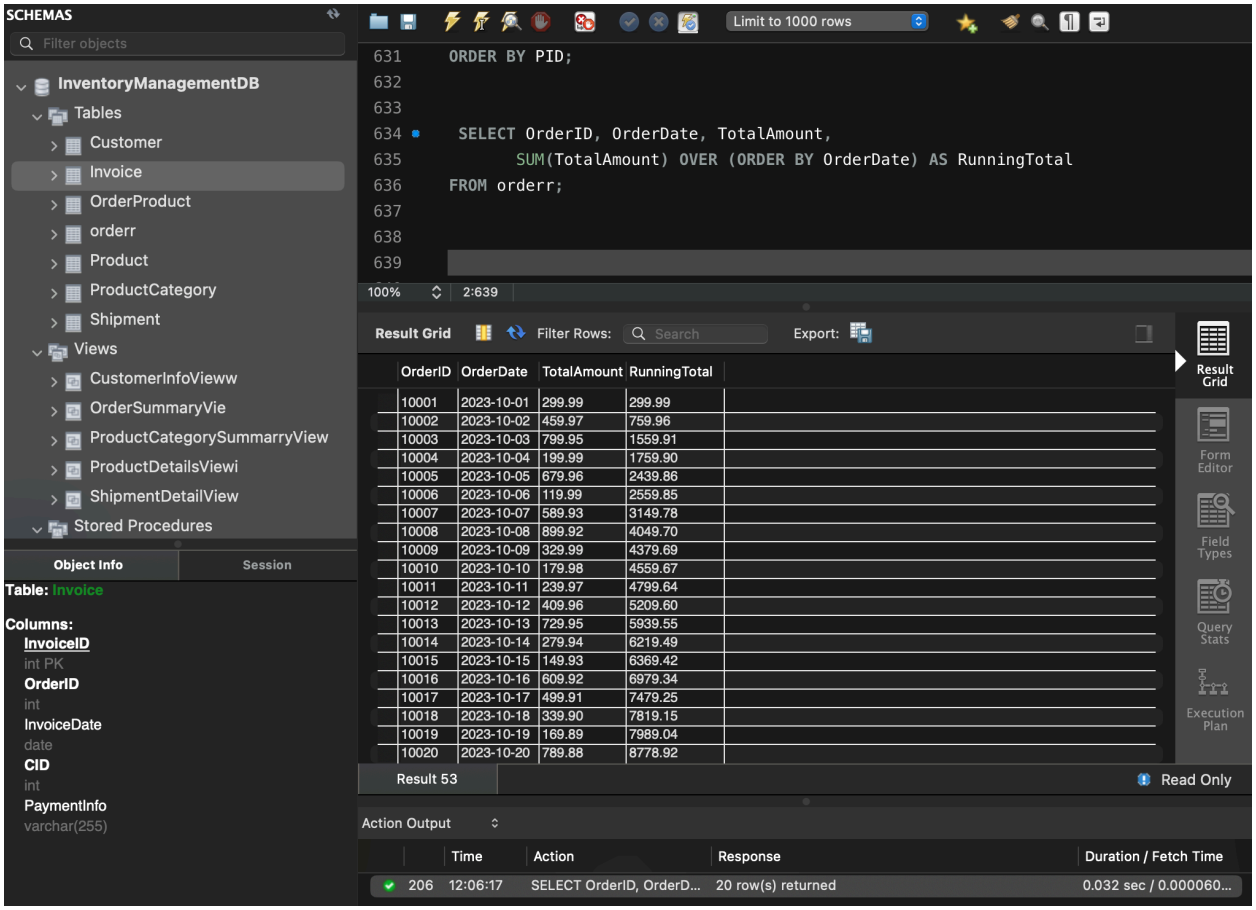
8) OLAP Function - Running Total of Orders:

Query:

```
SELECT OrderID, OrderDate, TotalAmount,
    SUM(TotalAmount) OVER (ORDER BY OrderDate) AS RunningTotal
FROM orderr;
```

Explanation: This query calculates the running total of orders over time, ordering them by order date.

Output/result:

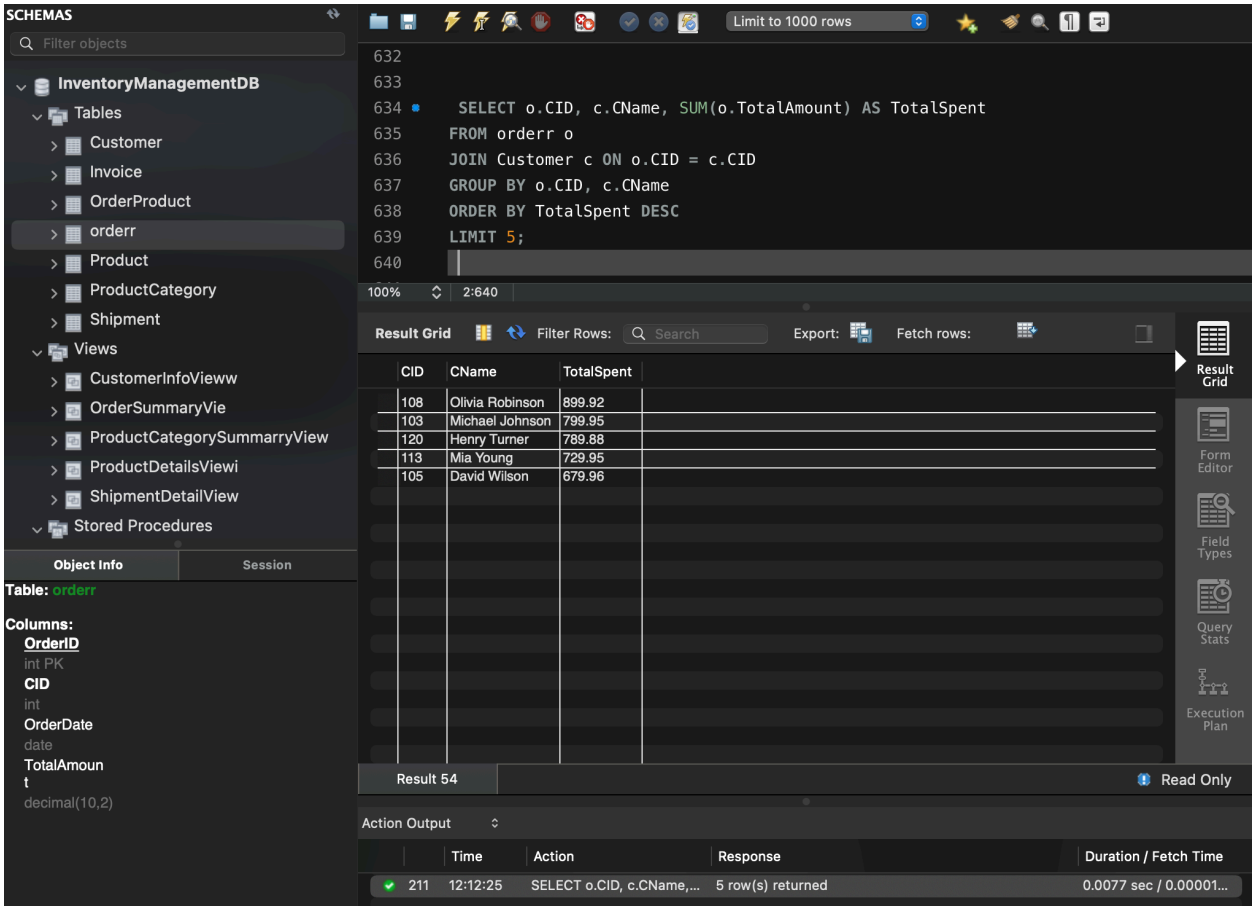


9) Top N Customers by Total Amount Spent:

```
SELECT o.CID, c.CName, SUM(o.TotalAmount) AS TotalSpent
FROM orderr o
JOIN Customer c ON o.CID = c.CID
GROUP BY o.CID, c.CName
ORDER BY TotalSpent DESC
LIMIT 5;
```

Explanation: This query identifies the top 5 customers who have spent the most based on the total amount in their orders.

Output/result:

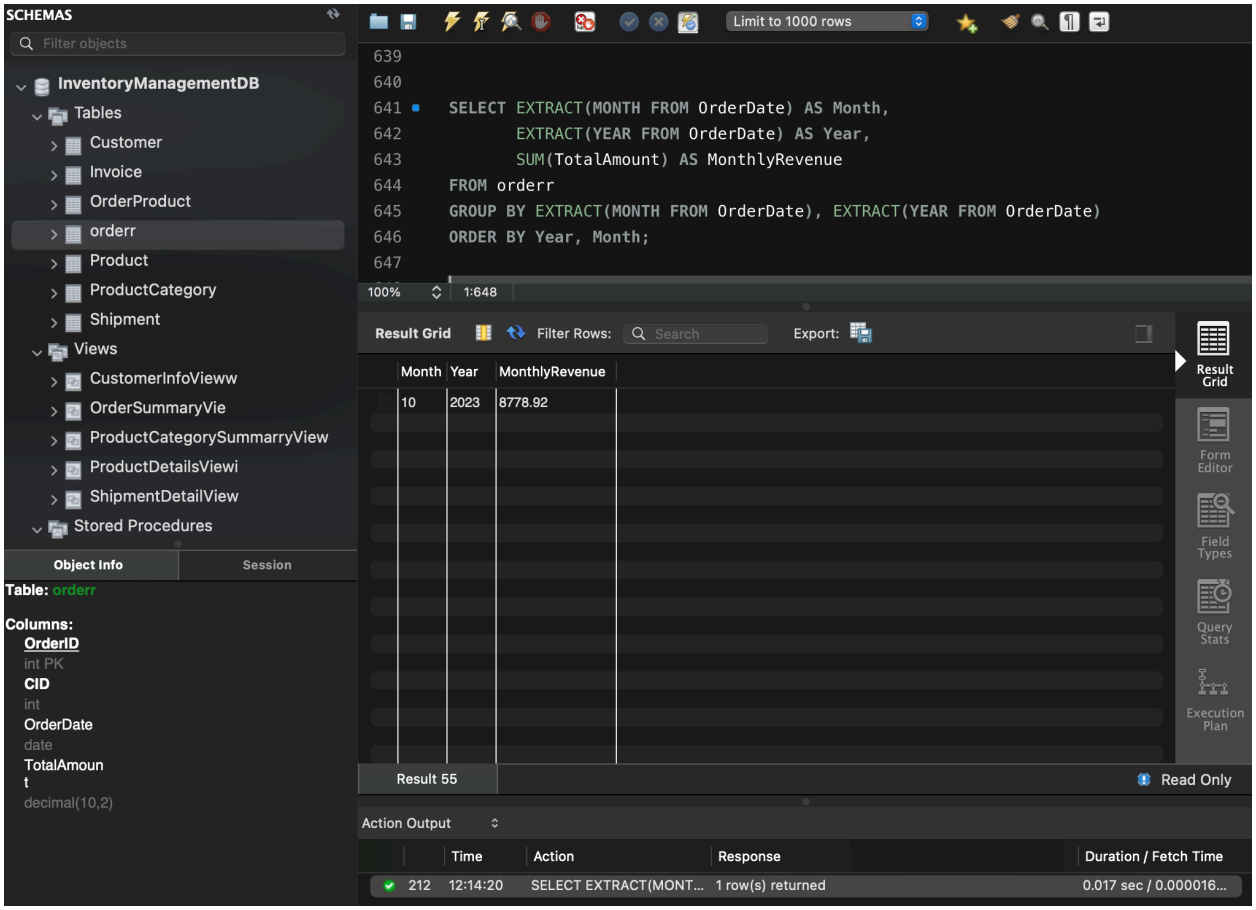


10) Total Revenue by Month:

```
SELECT EXTRACT(MONTH FROM OrderDate) AS Month,
    EXTRACT(YEAR FROM OrderDate) AS Year,
    SUM(TotalAmount) AS MonthlyRevenue
FROM orderr
GROUP BY EXTRACT(MONTH FROM OrderDate), EXTRACT(YEAR FROM
OrderDate)
ORDER BY Year, Month;
```

Explanation: This query calculates the total revenue for each month and year, providing a monthly breakdown of sales.

Output/result:



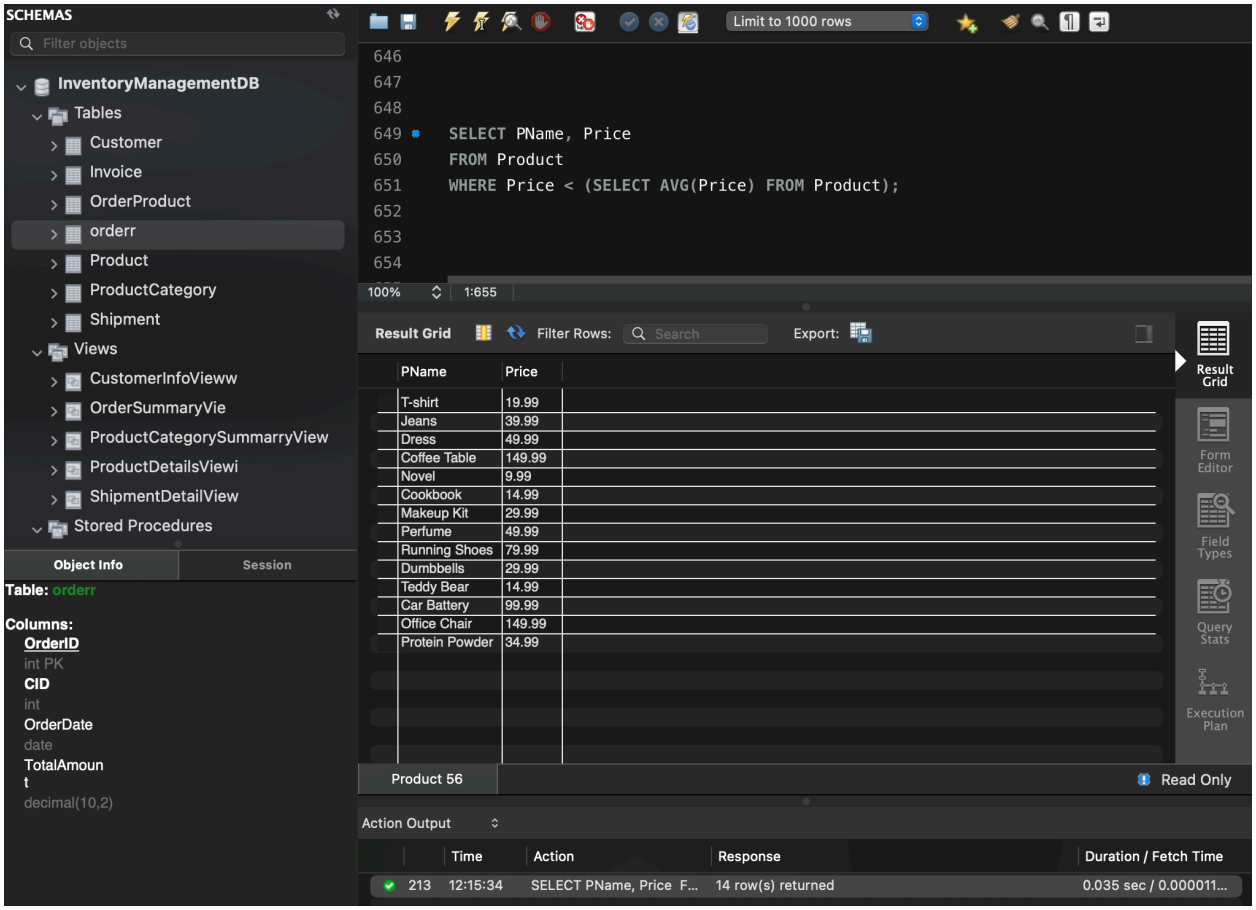
11) Subquery - Products Below Average Price:

Query:

```
SELECT PName, Price
FROM Product
WHERE Price < (SELECT AVG(Price) FROM Product);
```

Explanation: This query finds products with prices below the average price of all products in the Product table.

Output/result:



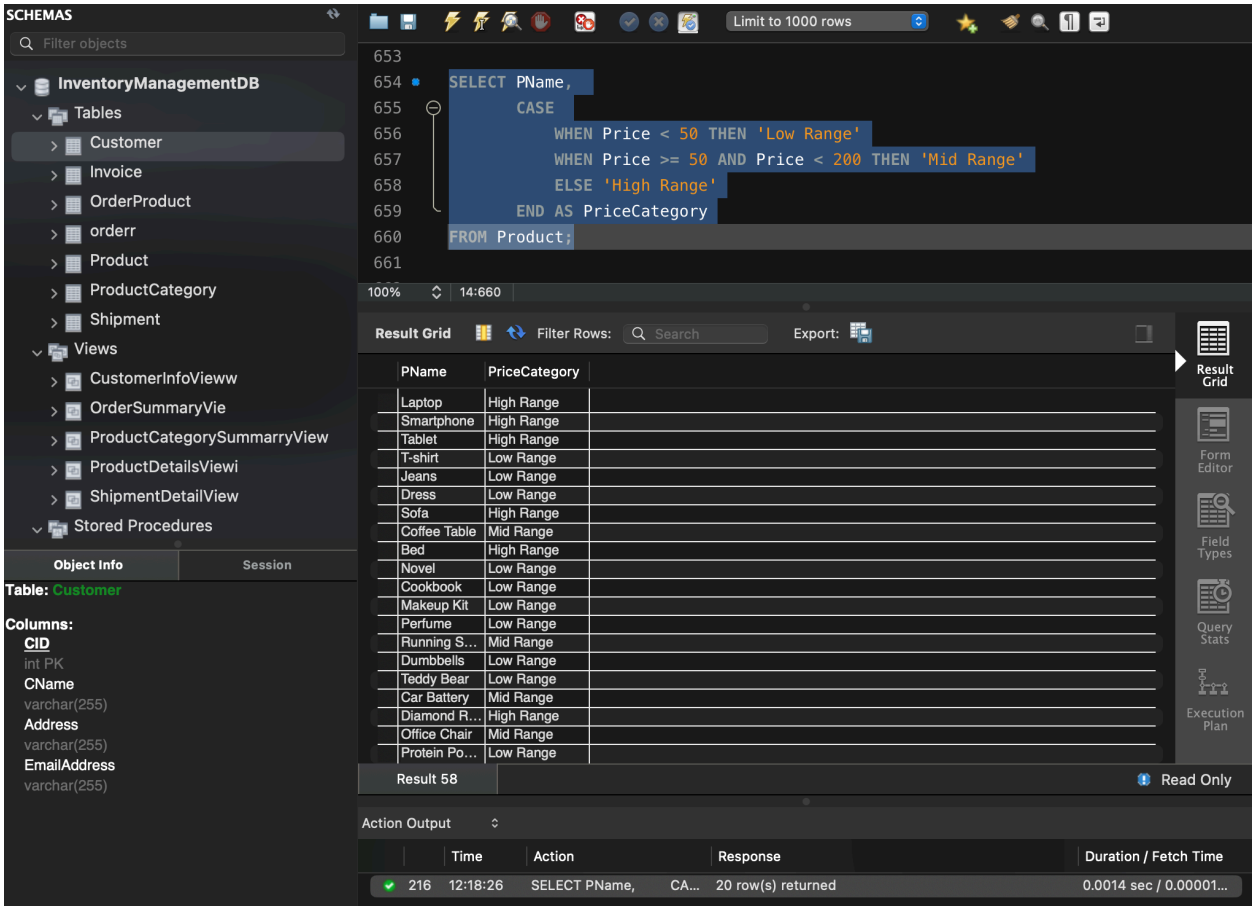
12) CASE Statement - Categorizing Products:

Query:

```
SELECT PName,
    CASE
        WHEN Price < 50 THEN 'Low Range'
        WHEN Price >= 50 AND Price < 200 THEN 'Mid Range'
        ELSE 'High Range'
    END AS PriceCategory
FROM Product;
```

Explanation: This query categorizes products into price ranges using a CASE statement.

Output/result:



13) DELETE Statement - Remove Low-Quantity Products:

Query:

```
DELETE FROM Product WHERE Quantity < 50;
```

Explanation: This query deletes products with quantities less than 50 from the Product table.

Output/result:

Before:

	PID	PName	SKU	Quantity	Price	Location	CategoryID
1	1	Laptop	LT1001	50	799.99	Warehouse A	1001
2	2	Smartphone	SP2002	100	499.99	Store 1	1001
3	3	Tablet	TB3003	30	299.99	Store 2	1001
4	4	T-shirt	TS4004	200	19.99	Warehouse B	1002
5	5	Jeans	JN5005	150	39.99	Store 1	1002
6	6	Dress	DR6006	80	49.99	Store 3	1002
7	7	Sofa	SF7007	20	499.99	Warehouse C	1003
8	8	Coffee Table	CT8008	40	149.99	Store 2	1003
9	9	Bed	BD9009	15	799.99	Warehouse A	1003
10	10	Novel	NV10010	300	9.99	Store 4	1004
11	11	Cookbook	CB11011	120	14.99	Store 3	1004
12	12	Makeup Kit	MK12012	80	29.99	Store 5	1005
13	13	Perfume	PF13013	150	49.99	Store 6	1005
14	14	Running Shoes	RS14014	90	79.99	Store 7	1006
15	15	Dumbbells	DB15015	50	29.99	Store 8	1006
16	16	Teddy Bear	TB16016	200	14.99	Store 9	1007
17	17	Car Battery	CB17017	30	99.99	Store 10	1008
18	18	Diamond Ring	DR18018	10	999.99	Store 11	1009
19	19	Office Chair	OC19019	25	149.99	Store 12	1010
20	20	Protein Powder	PP20020	100	34.99	Store 13	1011

After:

PID	PName	SKU	Quantity	Price	Location	CategoryID
1	Laptop	LT1001	50	799.99	Warehouse A	1001
2	Smartphone	SP2002	100	499.99	Store 1	1001
4	T-shirt	TS4004	200	19.99	Warehouse B	1002
5	Jeans	JN5005	150	39.99	Store 1	1002
6	Dress	DR6006	80	49.99	Store 3	1002
10	Novel	NV10010	300	9.99	Store 4	1004
11	Cookbook	CB11011	120	14.99	Store 3	1004
12	Makeup Kit	MK12012	80	29.99	Store 5	1005
13	Perfume	PF13013	150	49.99	Store 6	1005
14	Running Shoes	RS14014	90	79.99	Store 7	1006
15	Dumbbells	DB15015	50	29.99	Store 8	1006
16	Teddy Bear	TB16016	200	14.99	Store 9	1007
20	Protein Powder	PP20020	100	34.99	Store 13	1011

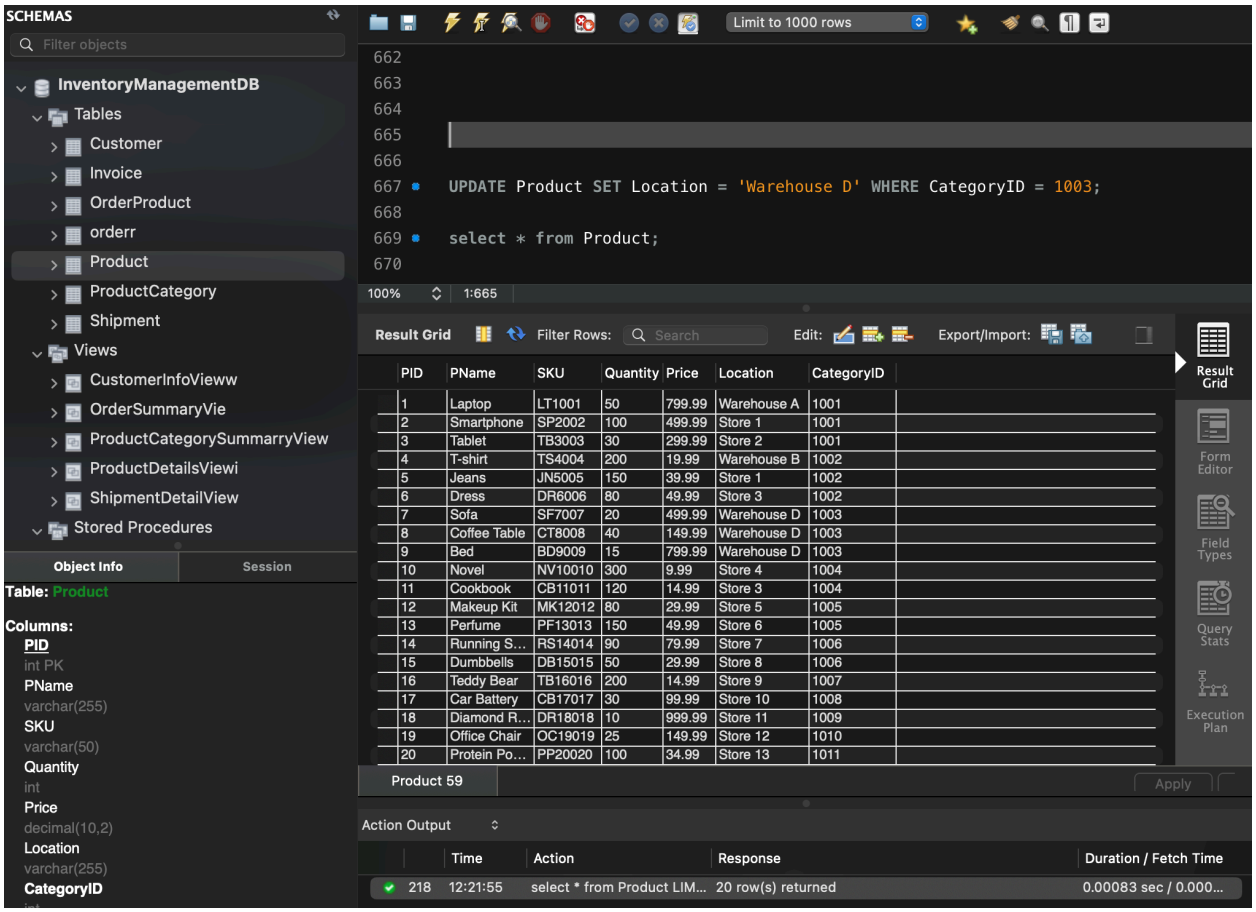
14) UPDATING Records - Change Product Location:

Query:

```
UPDATE Product SET Location = 'Warehouse D' WHERE CategoryID = 1003;
```

Explanation: This query updates the location of products in the 'Home and Living' category to 'Warehouse D'.

Output/result:



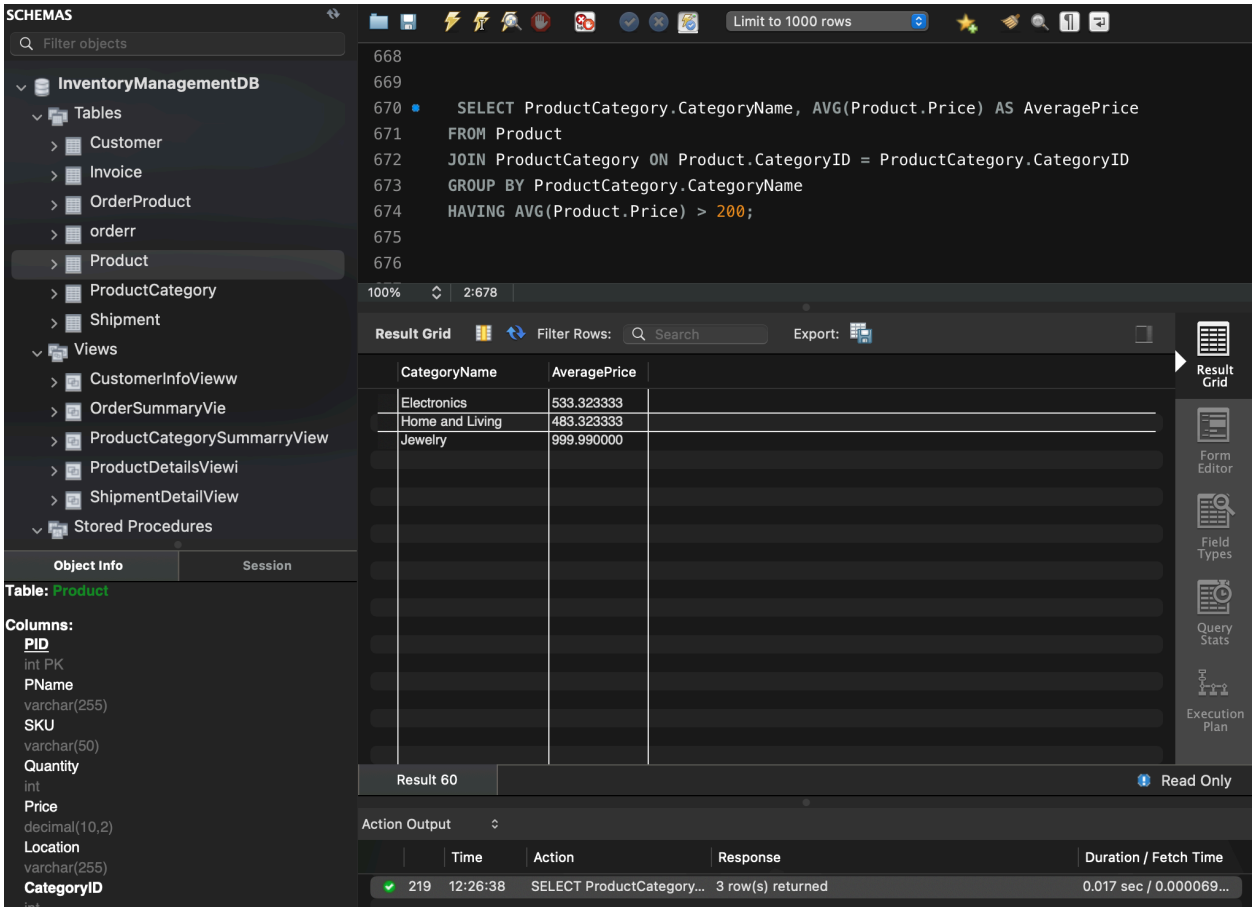
15) Group By with Having Clause - Categories with Expensive Products:

Query:

```
SELECT ProductCategory.CategoryName, AVG(Product.Price) AS AveragePrice
FROM Product
JOIN ProductCategory ON Product.CategoryID = ProductCategory.CategoryID
GROUP BY ProductCategory.CategoryName
HAVING AVG(Product.Price) > 200;
```

Explanation: This query finds categories with an average product price greater than 200, grouping the results by category name.

Output/result:



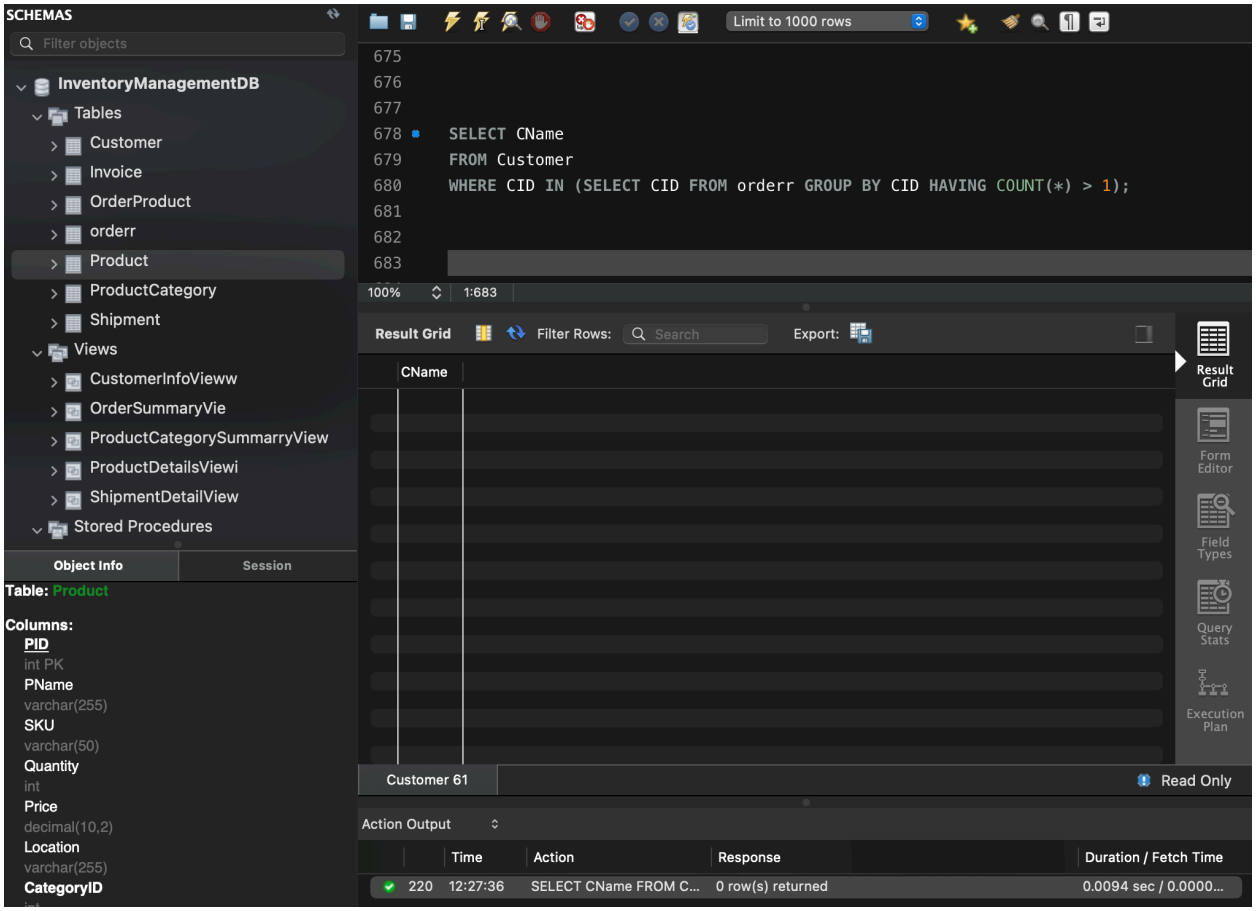
16) Nested Subquery - Customers Who Made Multiple Orders:

Query:

```
SELECT CName
FROM Customer
WHERE CID IN (SELECT CID FROM orderr GROUP BY CID HAVING
COUNT(*) > 1);
```

Explanation: This query finds customers who have made more than one order by using a nested subquery.

Output/result:



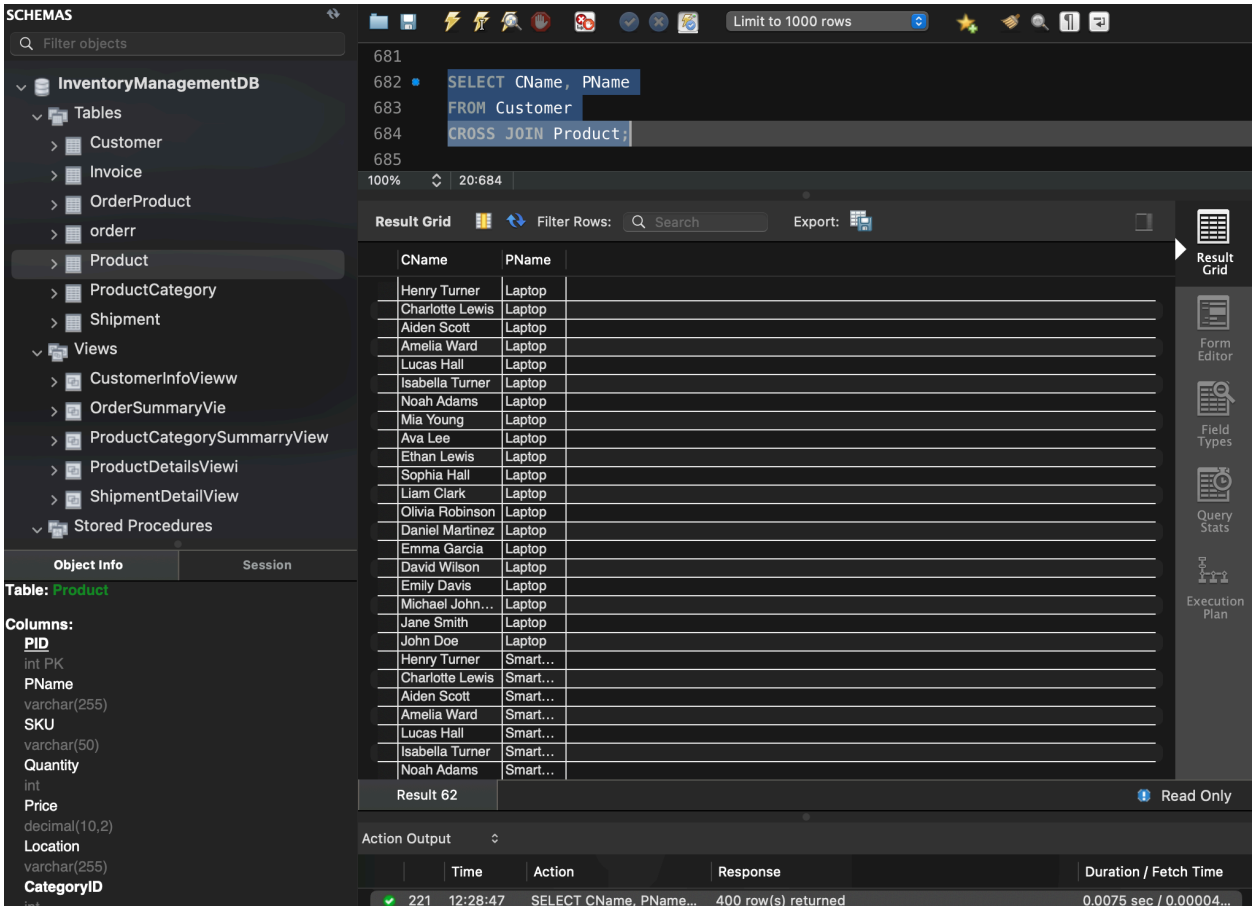
17) Cross Join - Cartesian Product of Customers and Products:

Query:

```
SELECT CName, PName
FROM Customer
CROSS JOIN Product;
```

Explanation: This query performs a cross join between Customer and Product tables, creating a Cartesian product of customers and products.

Output/result:


SCHEMAS
Filter objects
InventoryManagementDB
Tables
Customer
Invoice
OrderProduct
orderr
Product
ProductCategory
Shipment
Views
CustomerInfoVieww
OrderSummaryVie
ProductCategorySummarryView
ProductDetailsViewi
ShipmentDetailView
Stored Procedures
Object Info
Session
Table: Product
Columns:
PID
int PK
PName
varchar(255)
SKU
varchar(50)
Quantity
int
Price
decimal(10,2)
Location
varchar(255)
CategoryID
Limit to 1000 rows
681
682 SELECT CName, PName
683 FROM Customer
684 CROSS JOIN Product;
685
100%
20:684
Result Grid
Filter Rows:
Search
Export:
CName
PName
Henry Turner Laptop
Charlotte Lewis Laptop
Aiden Scott Laptop
Amelia Ward Laptop
Lucas Hall Laptop
Isabella Turner Laptop
Noah Adams Laptop
Mia Young Laptop
Ava Lee Laptop
Ethan Lewis Laptop
Sophia Hall Laptop
Liam Clark Laptop
Olivia Robinson Laptop
Daniel Martinez Laptop
Emma Garcia Laptop
David Wilson Laptop
Emily Davis Laptop
Michael John... Laptop
Jane Smith Laptop
John Doe Laptop
Henry Turner Smart...
Charlotte Lewis Smart...
Aiden Scott Smart...
Amelia Ward Smart...
Lucas Hall Smart...
Isabella Turner Smart...
Noah Adams Smart...
Result Grid
Form Editor
Field Types
Query Stats
Execution Plan
Result 62
Read Only
Action Output
Time
Action
Response
Duration / Fetch Time
221 12:28:47 SELECT CName, PName... 400 row(s) returned 0.0075 sec / 0.00004...